

**DAHLGREN DIVISION  
NAVAL SURFACE WARFARE CENTER**

Dahlgren, Virginia 22448-5100

---



**NSWCDD/TR-16/103**

**REVIEW OF KNOWLEDGE ENHANCED ELECTRONIC  
LOGIC (KEEL) TECHNOLOGY**

**BY ABID MEHMOOD  
GUNENDRAN SIVAPRAGASAM**

**TECHNOLOGY INTEGRATION SAFETY BRANCH (R44)**

**READINESS & TRAINING SYSTEMS DEPARTMENT**

**SEPTEMBER 2016**

DISTRIBUTION STATEMENT A - Approved for public release; distribution is unlimited.



<b>REPORT DOCUMENTATION PAGE</b>				<i>Form Approved</i> <i>OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. <b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b>					
<b>1. REPORT DATE (DD-MM-YYYY)</b> 12-09-2016		<b>2. REPORT TYPE</b> Technical		<b>3. DATES COVERED (From - To)</b> 1 January – 30 September 2016	
<b>4. TITLE AND SUBTITLE</b> Review of Knowledge Enhanced Electronic Logic (KEEL) Technology				<b>5a. CONTRACT NUMBER</b>	
				<b>5b. GRANT NUMBER</b>	
				<b>5c. PROGRAM ELEMENT NUMBER</b>	
<b>6. AUTHOR(S)</b> Mehmood, Abid Sivapragasam, Gunendran				<b>5d. PROJECT NUMBER</b>	
				<b>5e. TASK NUMBER</b>	
				<b>5f. WORK UNIT NUMBER</b>	
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)</b> Naval Surface Warfare Center, Dahlgren Division Technology Integration Safety Branch (R44) 5375 Marple Road, Suite 154 Dahlgren VA 22448				<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>  NSWCDD/TR-16/103	
<b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b>				<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>	
				<b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b>	
<b>12. DISTRIBUTION / AVAILABILITY STATEMENT</b>  Distribution Statement A – Approved for public release; distribution is unlimited.					
<b>13. SUPPLEMENTARY NOTES</b>					
<b>14. ABSTRACT</b> This report contains a review of the Knowledge Enhanced Electronic Logic (KEEL) technology, which provides a method of encoding expert knowledge for system control. KEEL allows a user to use drag-and-drop features of a Graphical User Interface (GUI) to reflect complex interrelationships between variables of a system. This allows the user to simulate the operation of the system in real time and fine-tune the relationships between inputs and expected system behavior. Once complete, KEEL can auto-generate code that can be inserted into applications to replicate the decision-making process of the expert. This review of KEEL consisted of analyzing the features of KEEL software and conducting testing to verify that the software accurately generated Java conventional code based on the design developed on the GUI.					
<b>15. SUBJECT TERMS</b> Knowledge Enhanced Electronic Logic, KEEL, Review, Testing, Verification, Analysis					
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b>  UL	<b>18. NO. OF PAGES</b>  34	<b>19a. NAME OF RESPONSIBLE PERSON</b> Gunendran Sivapragasam
<b>a. REPORT</b> UNCLASSIFIED	<b>b. ABSTRACT</b> UNCLASSIFIED	<b>c. THIS PAGE</b> UNCLASSIFIED			<b>19b. TELEPHONE NUMBER (include area code)</b> 540-653-7709

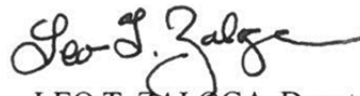
(THIS PAGE INTENTIONALLY LEFT BLANK)

## FOREWORD

This report contains a review of the Knowledge Enhanced Electronic Logic (KEEL) technology. KEEL, which provides a method of encoding expert knowledge for system control, allows a user to use drag-and-drop features of a Graphical User Interface (GUI) to reflect complex interrelationships between variables of a system. This allows the user to simulate the operation of the system in real time and fine-tune the relationships between inputs and expected system behavior. Once complete, KEEL can auto-generate code that can be inserted into applications to replicate the decision-making process of the expert or user. This review of KEEL consisted of analyzing the features of KEEL software and conducting testing to verify that the software accurately generated Java conventional code based on the design developed on the GUI.

Abid Mehmood and Gunendran Sivapragasam of the Naval Surface Warfare Center, Dahlgren Division (NSWCDD), Technology Integration Safety Branch (R44), prepared this document with the concurrence of Thomas M. Keeley, Compsim Limited Liability Company (LLC). This document was reviewed by Rebecca Sullivan, Head, Technology Integration Safety Branch (R44).

Approved by:

A handwritten signature in black ink, appearing to read "Leo T. Zaloga", with a stylized flourish at the end.

LEO T. ZALOGA, Deputy Department Head  
Readiness and Training Systems Department

(THIS PAGE INTENTIONALLY LEFT BLANK)

## CONTENTS

<u>Section</u>	<u>Page</u>
1 INTRODUCTION .....	1-1
1.1 PURPOSE.....	1-1
1.2 SCOPE.....	1-2
1.3 BACKGROUND .....	1-2
2 SYSTEM DESCRIPTION.....	2-1
2.1 SYSTEM OBJECTIVE .....	2-3
2.2 TYPES OF KEEL ENGINES.....	2-3
2.3 DEVELOPMENT HISTORY .....	2-4
2.4 SYSTEM CAPABILITIES.....	2-4
2.5 CONFIGURATION .....	2-5
2.6 USER INTERACTION .....	2-5
3 SYSTEM OPERATIONS.....	3-1
3.1 TRAINING .....	3-1
4 KEEL ASSESSMENT.....	4-1
4.1 ASSESSMENT APPROACH .....	4-1
4.2 SUMMARY OF ANALYSES AND TESTS PERFORMED .....	4-2
4.2.1 Thermostat Design.....	4-2
4.2.2 Arbitrary Design .....	4-3
5 CONCLUSIONS AND RECOMMENDATIONS .....	5-1
5.1 CONCLUSIONS .....	5-1
5.2 RECOMMENDATIONS.....	5-1
DISTRIBUTION.....	(1)

## ILLUSTRATIONS

<u>Figure</u>	<u>Page</u>
2-1 KEEL DESIGN FOR RISK OF A MISHAP DUE TO SPEED, TRAFFIC, AND WEATHER.....	2-1
4-1 THERMOSTAT DESIGN.....	4-2
4-2 DESIGN WITH ALL POSSIBLE KEEL FUNCTIONS .....	4-4
4-3 TEST APPLICATION RESULTS .....	4-5

(THIS PAGE INTENTIONALLY LEFT BLANK)

## GLOSSARY

ASCII	American Standard Code for Information Interchange
CMT	Compsim Management Tool
DGL	Dynamic Graphical Language
GUI	Graphical User Interface
IBIS	Issue-Based Information System
IDE	Integrated Development Environment
KEEL	Knowledge Enhanced Electronic Logic
LLC	Limited Liability Company
MAC	Macintosh
MIL-STD	Military Standard
NSWCDD	Naval Surface Warfare Center, Dahlgren Division
PC	Personal Computer
SME	Subject Matter Expert
US	United States
XML	Extensible Markup Language

(THIS PAGE INTENTIONALLY LEFT BLANK)

## **EXECUTIVE SUMMARY**

This review provides an overview of Knowledge Enhanced Electronic Logic (KEEL) technology and describes testing and analysis that was conducted on the KEEL Toolkit to determine if it accurately auto-generates Java conventional code that replicates models that have been designed in its Graphical User Interface (GUI). KEEL provides tools to develop portions of a system where decisions need to be made based on certain complex interrelationships and provides the capability for a Subject Matter Expert (SME) to design a model of these relationships that will provide the best decision based on the inputs provided. SMEs can use KEEL to test a model extensively before using KEEL to translate the model into conventional code.

This report addresses the ability of KEEL technology to translate these expert models into Java conventional code. Two models were developed and KEEL was used to translate both models into Java code. Simulators were used to provide inputs to the generated code, and the outputs of the code were compared to expected system behavior. Software analysis tools like PMD, FindBugs, and SonarQube were used to analyze code quality and investigate possible weaknesses in the Java conventional code that was generated. The results of the conducted analysis and testing showed that KEEL technology accurately generated Java conventional code that reflected the two chosen models without any errors.

(THIS PAGE INTENTIONALLY LEFT BLANK)

## 1 INTRODUCTION

Compsim Limited Liability Company (LLC) was founded in 1999, and the first product it developed was the Compsim Management Tool (CMT). CMT is a software application that runs on a Windows-based Personal Computer [PC] and provides users a process to facilitate the collection and validation of information in order to make rational and explainable decisions. CMT uses a process called Issue-Based Information System (IBIS) (conceived by Dr. Horst Rittel in the 1970s) on computers.

IBIS is a process that organizes data and targets “WICKED” problems – problems that are difficult or impossible to solve because of incomplete, contradictory, changing requirements and complex interdependencies that are often difficult to recognize. Dr. Rittel proposed that, if all issues, positions, and arguments on any given topic were organized, then it would be easier to make the best decision. Additionally, he asserted that “WICKED” problems are too complex to be solved by writing a formula to produce a correct solution.

The IBIS process was initially executed on notecards. Compsim LLC implemented the IBIS process on computers in the form of the CMT product, which organizes information in a knowledge tree. A human then uses the knowledge tree to make a decision. Knowledge Enhanced Electronic Logic (KEEL) was developed as the next step, to use the information in the knowledge tree to make the most optimized decision.

A KEEL model was first used for a software-based weapon system. It was quickly discovered that the initial KEEL model did not have much flexibility, addressing only one structured problem at a time (as was addressed with CMT). KEEL was then expanded to address multiple problems at a time that had interrelated and sometimes conflicting characteristics. Compsim LLC introduced KEEL wires to connect variables together and define the relationships between those variables. The wires defined a functional relationship type and, over time, more wires were introduced to define other needed functional relationships. This allowed the user to build a model with complex webs of interrelationships. The three main characteristics of the current KEEL can be summarized as below:

- A developmental environment
- A tool to create a model for accumulating supporting and objecting arguments in order to make a decision or take an action
- A small footprint engine that processes sensors’ data or other inputs according to the design of a system created in the development environment

### 1.1 PURPOSE

This report has been developed to support a Cooperative Research and Development

Agreement, a joint effort between the Naval Surface Warfare Center, Dahlgren Division, and Compsim LLC. The purpose of this report is to provide a review of the KEEL technology, providing an independent analysis on whether there are any hurdles to using KEEL technology in United States (US) Navy applications. It should be noted that this review is focused only on the ability of KEEL technology to translate expert models into Java conventional code. KEEL technology is an already developed and mature technology and is not a milestone-driven program.

## 1.2 SCOPE

The main purpose of this technology is not to provide tools to develop an entire system but just portions of the system where decisions need to be made based on certain complex interrelationships. When this portion is integrated into a complete system, the user still has to develop an application package and write methods to provide input into the KEEL Engine and receive output from it. KEEL was developed so complex (dynamic, nonlinear, interrelated, multidimensional) models could be developed by a Subject Matter Expert (SME) without dependence on the use of higher level mathematics (e.g., predicate calculus) and to avoid long development and debug cycles (because the code is always the same). With KEEL, the objective is that the complex models can be developed and initially tested by the SME before ever translating the models into conventional code.

This report addresses the ability of KEEL technology to produce a KEEL Engine from a model developed in the KEEL Toolkit for one of the languages supported by KEEL, Java, through analysis and in-depth safety testing. Software analysis tools like PMD, FindBugs, and SonarQube have been used to ensure code quality and point out possible errors in the KEEL Engine. No assessment was done on the KEEL Toolkit that is responsible for generating the KEEL Engine. In summary, in-depth testing verified that the KEEL Toolkit generates the intended design or KEEL Engine in Java. However, it is still recommended that users of this technology developing safety significant code should perform software safety analysis per the prescribed Military Standard (MIL-STD)-882E software safety analysis procedure. The appropriate level of rigor should be applied to the code that is generated by the KEEL Toolkit to minimize the possibility of a mishap.

## 1.3 BACKGROUND

KEEL has been around for more than a decade by now and is currently used in many different industrial applications. This technology has potential use in fields like Modeling and Simulation where it can provide Rapid Development Cycle capability, an accurate representation of target system behavior, and improvements in life-cycle cost management. Also, KEEL is very useful when dealing with nonlinear complex problems where it is very difficult to predict behavior using mathematical formulations, and where relationships between components are very dynamic. Finally, KEEL could potentially be used in military applications, e.g., in autonomous systems (Unmanned Autonomous Vehicles, Unmanned Ground Vehicles), where these systems make decisions based on given conditions such as the environment, situation, and priorities.

## 2 SYSTEM DESCRIPTION

The KEEL development environment is very different from an Integrated Development Environment (IDE) of a conventional programming language. All components of KEEL become active parts of the system as soon as a user drops them in the development environment. All components work simultaneously, which is a big shift in the thinking process as compared to a conventional programming language that usually has a sequential flow. Because of that, input validation (if input A=Input B, then execute ...) can't be implemented. Despite this, KEEL Engines will function correctly as components of a system coded in a conventional way. The KEEL Engine is called when complex decision-making (system judgment and reasoning) is required.

KEEL allows human decision-making to be replicated in software using knowledge from SMEs. The experience and know-how of these SMEs are captured in the KEEL Toolkit using supporting and objecting arguments in order to make a decision or to take an action. An example of a decision-making structure using the KEEL Toolkit can be seen in Figure 2-1 below:

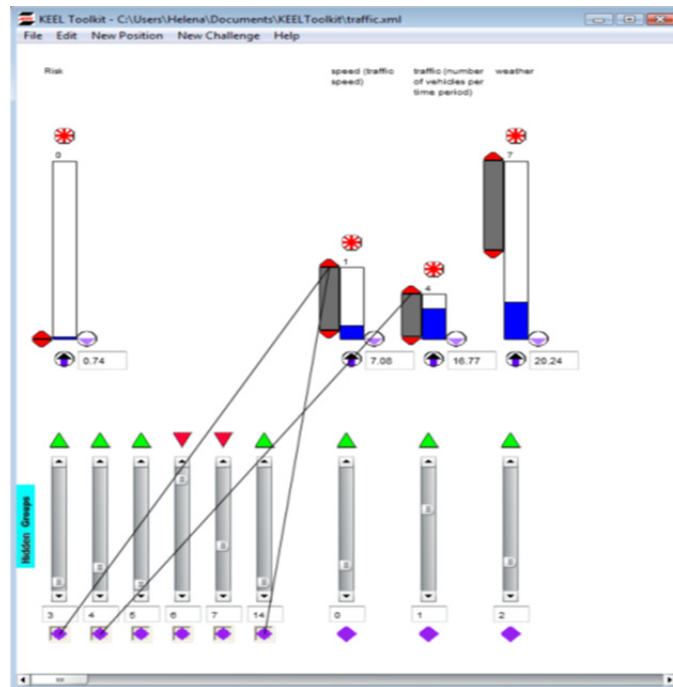


FIGURE 2-1. KEEL DESIGN FOR RISK OF A MISHAP DUE TO SPEED, TRAFFIC, AND WEATHER

The KEEL Dynamic Graphical Language (DGL) is the actual “KEEL Source Code” in the KEEL Toolkit. The KEEL DGL makes use of the interactive nature of computer graphics to create the KEEL Engine. In the figure above, icons representing elements of the KEEL DGL are dropped on the screen and linked together by wires representing specific functional relationships

between data items. The user stimulates the system (through slider icons) and observes the system response.

The design above uses variables like speed, traffic, and weather to determine the risk of a mishap. The vertical scroll bars with green and red indicators at the bottom of each of the actions (positions) are supporting and blocking inputs. The wires in the diagram show that different actions and/or inputs can be linked together to affect actions.

Once an SME has created a design, this graphical source code can be translated to a conventional code (e.g., C, C#, C++, JAVA, FLASH, Visual Basic, etc.). The KEEL Engine is produced in the form of a text (.txt) file that is copied and pasted into the user's development environment to be compiled.

Two KEEL Engine processing models are available for most languages: The "Normal Model" processes information as if it was processed on an analog computer (thus the high Cyclomatic Complexity discussed further in Section 4.2.2). Inputs and outputs are balanced until a stable set of inputs and outputs is achieved. The "Accelerated Model" is created after the KEEL Toolkit calculates an optimal processing path. The Accelerated Model contains one additional table to control the processing order. The user selects the most appropriate model for the target application.

The KEEL Engine takes the data in the design and stores it in tables that define the values and relationships of the inputs and actions. The code then creates arrays in which this data is stored and manipulates the data to reflect the design.

The artifacts of the KEEL Engine are the KEEL "dodecisions" function and the KEEL "tables" (1- and 2-dimension arrays; or 1 multidimensional array for the Accelerated Code). Depending on the conventional programming language selected, there may be an initialization routine (initializefixedtables) or a constructor if an object-oriented language is selected.

In summary, a KEEL Engine includes code to process tables of information in a consistent way (all KEEL Engines will be the same). The "code" is approximately 381 lines of Normal version uncommented code (for C). While this is conventional code, it is of no use in understanding the entire problem being solved, because the problem definition is maintained in the arrays, not the code. The code manipulates the data in the arrays according to the design that was created in the KEEL Toolkit.

The objective is that the 381 lines of code will only need to be validated one time, because the code will always be the same. Following are three examples of how the KEEL code has been used for applications written in C:

1. Obstacle analysis and collision avoidance code for a land robot: 381 lines of C code

2. Analysis of the results of a hematological analyzer to determine the type of anemia: the same 381 lines of C code
3. Assessment of a target moving through varying terrain to determine the instant to shoot written in C: the same 381 lines of C code

## 2.1 SYSTEM OBJECTIVE

KEEL includes a graphical language with vertical scroll bars; up/down arrows; upward arrows in circles; and lines representing positions, supporting/objecting arguments, thresholds, and relationships between them. These icons represent “information items” and “connection points” with which to define functional relationships (rather than scripting “code” in the conventional sense).

KEEL allows users to make human-like decisions based on the arguments, thresholds, relationships, and priorities. It allows users to model very simple linear to very complex behavior, which could be very difficult to capture with a mathematical formulation or other computer program. The graphical language also makes it easy to “see” the functional relationships and the dynamic (interactive) nature of the language, allows one to interact with a design and see the impact of the relationships as they are being defined.

Once the design is complete, a KEEL Engine is created by the KEEL Toolkit – conventional code can be generated for the design in any language of choice in text format. The user has the choice of creating either unoptimized (Normal Type) code or optimized (Accelerated Type) code.

## 2.2 TYPES OF KEEL ENGINES

KEEL technology was originally developed to be used in embedded systems. Embedded systems sometimes have additional considerations such as:

- Smallest possible memory space
- Fastest possible operation and to be more deterministic (as far as processing time)

To achieve these two goals, two different types of KEEL Engines can be created. There is very little difference between the two types. They are labeled Normal and Accelerated. The Normal code is slightly smaller, and slightly slower (in most cases), with slightly more execution cycles of non-used functionality. The Accelerated code is slightly larger, slightly faster (in most cases), with slightly fewer execution cycles.

The difference between the two versions is that the Accelerated code includes one additional multidimensional table that defines an optimal way to process the information. This takes more memory space but accelerates the processing (in most, but not all, cases).

## 2.3 DEVELOPMENT HISTORY

Soon after Compsim LLC started in 1999 with its first product – CMT, Thomas Keeley and Helena Keeley came up with the idea of KEEL upon realizing the patented CMT information fusion model (US 6,833,842) could be used by machines to make more complicated, judgmental types of decisions. KEEL was leveraged off CMT, which was a tool developed to organize information and to make and explain “subjective decisions” (e.g., feature selection for a project, risk analysis, vendor selection, project prioritization).

During initial tests, when applying the CMT model to an autonomous weapon system, there was recognition that an autonomous robot did not have the luxury of solving one problem at a time. It had its primary objective; a number of intermediary goals; the need to change goals; the need to allocate resources across multiple goals; the need to manage its own survival; the potential for operation under degraded capabilities; and the need to monitor its environment and to change “everything.” So rather than CMT’s structured problem solving, KEEL evolved to its present state of addressing webs of interrelationships that have to be addressed “collectively.” A series of KEEL-related patents has since been applied for and granted.

The first three KEEL patents were filed on October 2002, with application numbers US 10/289,663, US 10/289,517, and US 10/289,477, resulting in granted patents: 7,039,623 (5/2/06), 7,009,610 (3/7/06), and 7,159,208 (1/2/07). These covered the information fusion model, the method for processing information in the KEEL Engine, and the DGL. In October 2004, application number US 10/960,626 covered a KEEL Engine that could be released in silicon (without a microprocessor) and also served for the Accelerated Processing Model (Patent number 7,512,581 (3/31/2009)). In June 2006, application US 11/446/801 was submitted to support additional functional relationships with graphics and processing. Patent 7,685,528 was granted on 3/23/2010. On March 2014, application US 14/202,736 was submitted to handle additional inverted functional relationships with graphics and processing.

## 2.4 SYSTEM CAPABILITIES

KEEL Engines are delivered as functions or class methods in many source code languages including C, C++, C#, Objective C, DART, Flash, HAXE, Java, Java Script, Octave, Python, SCILAB, and Visual Basic with many different “wrappers” for different development environments. KEEL Engines can be compiled and then executed by most digital micro-controllers and computers in existence today.

KEEL makes it easy to capture complex SME knowledge using the KEEL DGL and then producing conventional code for a KEEL Engine that replicates that knowledge. The KEEL Toolkit Graphical User Interface (GUI) is very simple and easy to learn. Any dropped component in KEEL’s development environment becomes active and part of the overall system instantly.

The KEEL GUI also enables the user to see the functional relationships between components of a KEEL Engine. A user can create relationships of all types and sorts. The user

considers how pieces of information combine to influence other pieces of information, drags a wire between connection points, and observes the functional relationship in operation, getting immediate feedback.

## 2.5 CONFIGURATION

The KEEL Toolkit can be used on the PC and Macintosh [MAC] platforms. The KEEL Engines (created by the KEEL Toolkit) do not have any special hardware requirements and can be run on any microcontroller or commercial computer with basic hardware. No external libraries are required for the basic KEEL Engine when inserted into the user's application. It should be noted that there is a significant advantage to using a system with multiple, large displays when working with the KEEL development environment (KEEL Toolkit). A single display (1024x768) is an absolute minimum.

Additionally, an internet connection with a fixed IP address is needed for the Evaluation License validation and access to on-line training. There are many factors that determine the scope of the production license (e.g., individual product/application, product line, market area, exclusive/nonexclusive, royalty/fixed price) and this should be discussed directly with Compsim LLC.

## 2.6 USER INTERACTION

For user interaction, a GUI is provided for the KEEL Toolkit as mentioned above. A user can start a new project using the 'File' menu from the list. The "New Position" and "New Challenge" menu items are used to drop positions and challenges of certain values and types. Using mouse clicks, a user can create wires to create relations between components of the KEEL Engine. When a user creates a KEEL Engine in the KEEL environment, an Extensible Markup Language [XML] file is created behind the scenes that can be used to pull up the same design the next time it is needed. Also using the 'File' menu, a user can select the conventional code of choice. As mentioned earlier, no study has been conducted to investigate how this code is created from an XML file/graphical source code to a text file. For this report, a KEEL Engine was produced in Java to conduct testing to verify that the KEEL Toolkit produces the intended KEEL Engine.

(THIS PAGE INTENTIONALLY LEFT BLANK)

### **3 SYSTEM OPERATIONS**

#### **3.1 TRAINING**

Compsim LLC offers a variety of training, including a Web-based training course consisting of 11 modules, which walks a trainee step by step through various topics. Here, each component of KEEL is explicitly explained with examples. Also a quiz is offered at the end of each module to quickly check the knowledge gained from the module. Additionally, a training manual has been developed with practical examples. The manual is more advanced than the 11 basic Web-based modules and is available for download from the KEEL menu. Lastly, Compsim LLC also offers face-to-face code walk-through training.

(THIS PAGE INTENTIONALLY LEFT BLANK)

## 4 KEEL ASSESSMENT

In this review, KEEL technology was assessed by choosing two designs to model and generate KEEL Engines. The generated KEEL Engines were assessed using in-depth testing to validate that the KEEL Engine in conventional code had the same functionality that was designed in the KEEL development environment. Also, line-by-line code analysis and code analysis using static code analysis tools were performed to capture any defects in the code. No assessment has been done on the KEEL DGL or Toolkit code that produces the conventional code for any particular design made utilizing the KEEL technology. It is recommended that thorough safety analysis of any safety-significant function created using KEEL technology be conducted prior to employment of that KEEL Engine.

### 4.1 ASSESSMENT APPROACH

The KEEL Toolkit is assumed to be consistent in creating KEEL Engines in multiple languages (e.g., Java, Visual Basic, C++).

The 381 lines of KEEL code (for C) will always remain the same for all Normal engines created by the KEEL Toolkit<sup>1</sup>. It is only the arrays that are different based on the design created by the user in the KEEL Toolkit. As such, analysis was conducted to determine that a KEEL Engine with every possible feature or function that can be created in one language (Java) does the following:

- Creates the data tables or arrays correctly
- Manipulates the data in those arrays correctly

This would provide confirmation that, in a limited example, the KEEL Engine created correctly implements the developer's design in the KEEL Toolkit.

Note: Users using the KEEL Toolkit to develop safety-significant code should conduct software safety analysis utilizing the process outlined in MIL-STD-882E. The appropriate level of rigor should be applied to the code that is generated by the KEEL Toolkit to minimize the possibility of a mishap.

The KEEL Toolkit was reviewed using the following approach:

- a) Due to limited resources, it was assessed utilizing one language – Java.

---

<sup>1</sup> A user option is available to optimize the small KEEL Engine code segment to eliminate code that is not required.

- b) First, KEEL models were created using KEEL features and functions.
- c) Then the 'Snap' feature was used to log the inputs and outputs from the KEEL Toolkit. The Snap 'feature' is a menu item in the KEEL Toolkit that generates an XML file (or snapshot) of all output and input variables. This allows an analyst to ensure that the model is performing as intended, generating the right outputs based on the inputs provided.
- d) Finally, the engine that was created (text file) was analyzed to see if the tables created and the handling of the data in those tables was equal to the design seen in the KEEL Toolkit.

This is a process similar to what is used by the developer to verify proper creation of KEEL Engines when a new programming language is added to the KEEL Toolkit. In fact, while this is the first time an independent review of the KEEL Toolkit has been performed, the process and methodology used are the same as what the developer used earlier to verify the implementation of the Java language in the KEEL Toolkit.

## 4.2 SUMMARY OF ANALYSES AND TESTS PERFORMED

### 4.2.1 Thermostat Design

A KEEL Thermostat design was used as an example to illustrate the above methodology to verify the code. Figure 4-1 below is a Thermostat design snapshot taken from the KEEL Toolkit environment. This design was made very simple to illustrate the behavior of the KEEL Engine for this design.

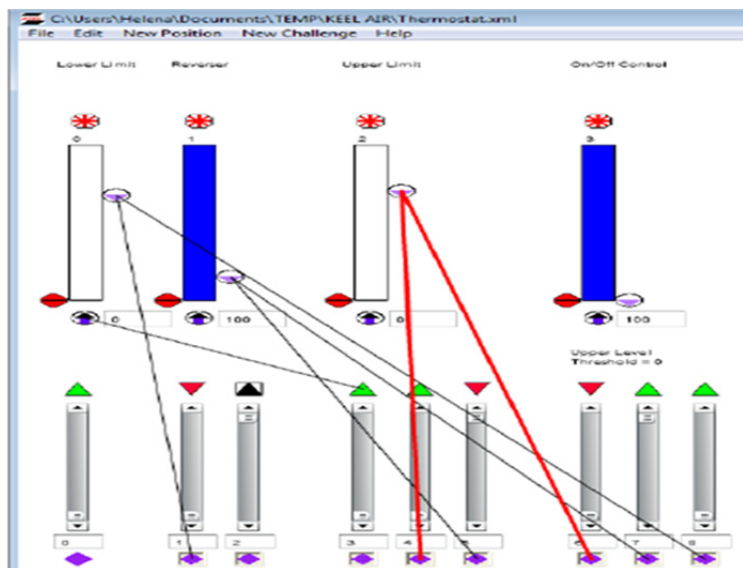


FIGURE 4-1. THERMOSTAT DESIGN

This design takes only one input from the user (sensor's temperature) and provides a discrete output to turn on/off the furnace. In this design there are two thresholds, a lower threshold and an upper threshold, which are set at 67 and 70 respectively. Note that these thresholds can also be made variable, which allows the user to change them dynamically, but again, these arguments were fixed for simplicity. The wires between the variables are the functional relationships of those variables.

Initially, the input was set at 0 and the output was set at 100, which indicates a sensor input of 0 degrees Fahrenheit resulting in the Thermostat turning on the furnace (output at 100). When the input is increased from 0 to 70 (which is the upper threshold), the output stays at 100, which indicates that the Thermostat keeps the furnace on. If the sensor input is increased any further than 70 (say 70.01), the output goes to 0 turning off the furnace. Now if the sensor input is decreased, say from 70.01 to 68, the output stays at 0 because it is still greater than the lower threshold. If the sensor input is decreased further to 66 (below the lower threshold), the output jumps once again to 100 (furnace turns on).

For the Thermostat design, Java code was generated in the text format. To verify this code, the text file was copied into a Java class using a NetBeans IDE. To interact with this class, another main Java class was written to insert inputs into the KEEL Engine and to receive outputs from it. In this case, there was only one input for the temperature and one output for furnace on/off. The above test case scenarios conducted in the KEEL Toolkit were repeated in the NetBeans IDE and the results were the same.

#### 4.2.2 Arbitrary Design

The previous example was limited and the full functionality of the KEEL Toolkit was not expressed in the design. To address this, an arbitrary design with every function that is available in the KEEL Toolkit was developed. A snapshot of this design from the KEEL environment is shown in Figure 4-2 below.

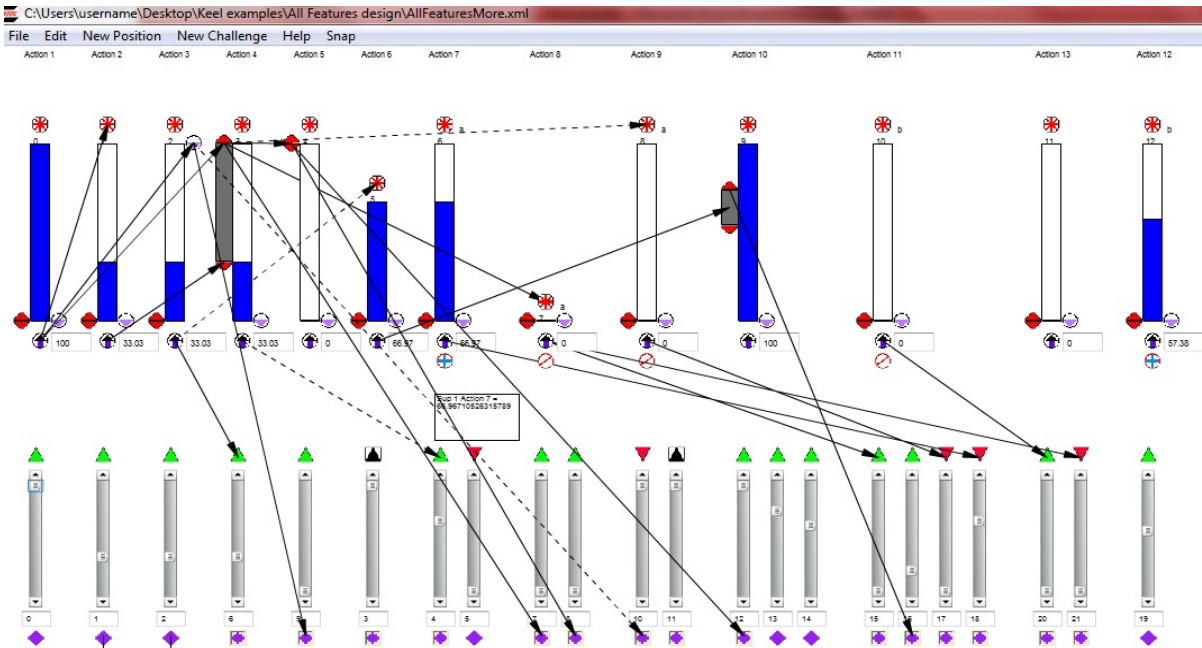


FIGURE 4-2. DESIGN WITH ALL POSSIBLE KEEL FUNCTIONS

This design is more complex than the previous Thermostat design and a “Test Application” built by Compsim LLC was used to verify/validate the design. This “Test Application” is a Java swing application that takes inputs and outputs from the XML design files, which are captured by the ‘Snap’ function of KEEL, and then inserts those inputs into the KEEL Engine (Java code generated for the design) and extracts the outputs. The “Test Application” then takes these outputs from the KEEL Engine and compares them against the outputs obtained from the XML file.

Figure 4-3 is a snapshot taken from the NetBeans IDE environment for the “Test Application.” The first column in the snapshot shows the inputs taken from the XML file that are used as inputs to the KEEL Engine in the NetBeans IDE for Java. The second column contains the outputs taken from the XML file. The third column notes the differences between the outputs from the XML file and the outputs from the KEEL Engine.

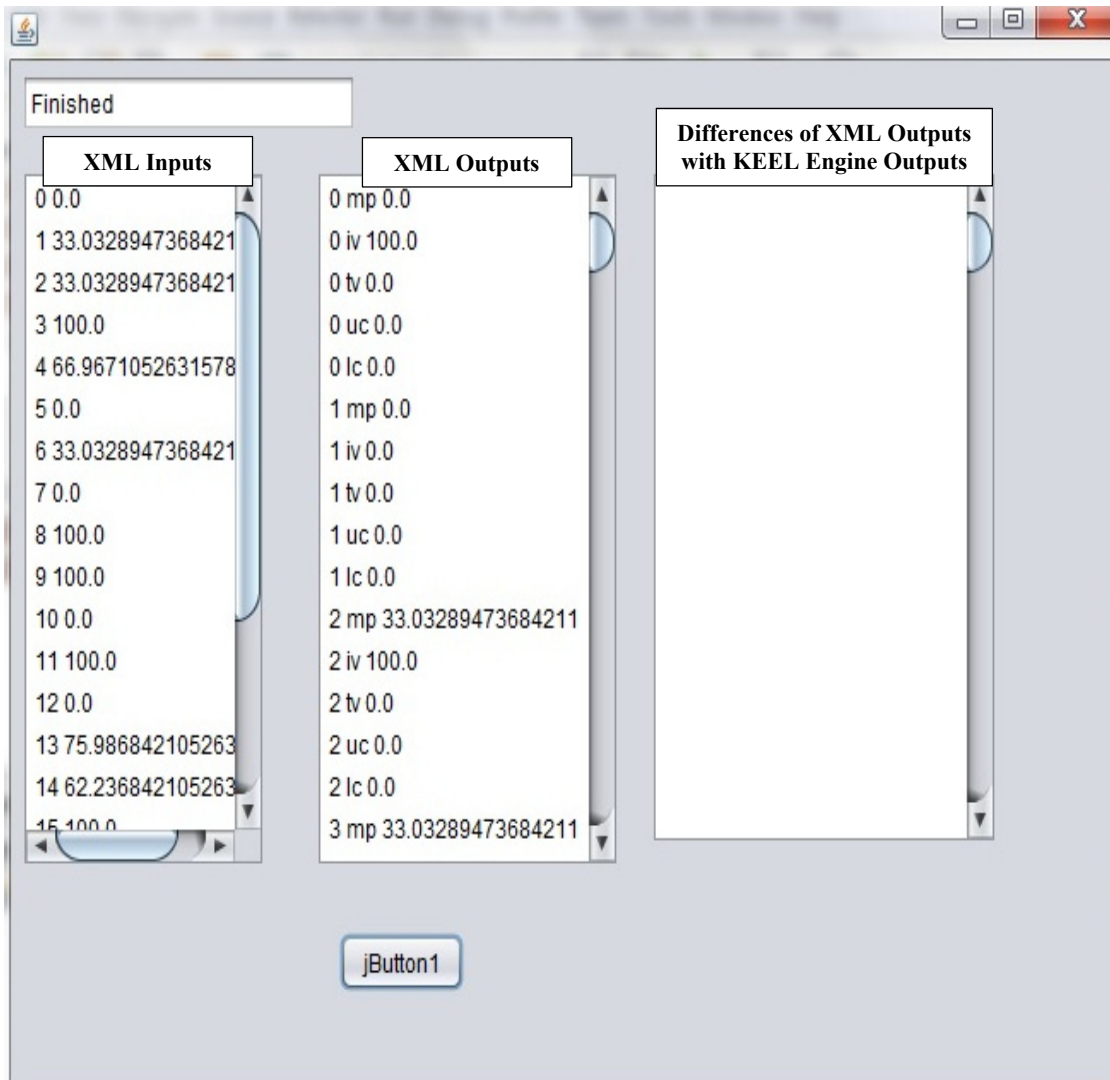


FIGURE 4-3. TEST APPLICATION RESULTS

In this case, the third column is empty because those two systems perform identically. It is possible to have some differences based on how different languages are compiled and the fact that floating point numbers are being sent from the KEEL Toolkit to the Test Application as American Standard Code for Information Interchange [ASCII] strings. The differences (if they are detected) usually are in the range of the fourteenth decimal place.

Additionally, a line-by-line code review was done to find possible bugs (e.g., infinite loops, uninitialized variables, incorrect array lengths, inaccessible code). No bugs were found during the manual code review.

For further confidence, “FindBugs,” “PMD,” and SonarQube static code analysis tools were used to find bugs in the code that are difficult to detect with manual code review (e.g.,

overflows, integer truncation, race conditions). Again, no bugs were found using these tools. However, the tools did provide suggestions on how the code could be further improved. As an example, the code has many nested loops, which results in high Cyclomatic Complexity. In this case, the Cyclomatic Complexity of the dodecisions method was found to be 92, which is higher than the complexity of 15 recommended by the Motor Industry Software Reliability Association. While maintaining a low Cyclomatic Complexity is good practice, there is no indication that the nested loops in the KEEL Engine introduce weakness in the code.

## **5 CONCLUSIONS AND RECOMMENDATIONS**

### **5.1 CONCLUSIONS**

No review of the KEEL Toolkit code that creates KEEL Engines has been performed. Rather, it has only been verified that Java source code in the text file created by KEEL (known as the KEEL Engine) contains code that correctly defines the data tables (arrays) and correctly manipulates the data in those tables (“dodecisions” function) according to the KEEL DGL model developed in the KEEL Toolkit. No issues were detected in the review of the KEEL Toolkit or example-generated Java code that would inhibit the use of KEEL in U.S. Navy applications.

### **5.2 RECOMMENDATIONS**

Listed below are recommendations for Compsim LLC and users of the KEEL technology based on the review conducted:

- a) Ensure that there is version control for the KEEL Toolkit and specifically for the creation of engines in each language.
- b) A new user to KEEL should use the code verification process defined in the Compsim LLC White Paper – “Certifiable Code and KEEL Technology” to ensure that the engine built by the KEEL Toolkit properly reflects the intended design of the user.
- c) Those using the KEEL Toolkit to develop safety-significant code should conduct software safety analysis utilizing the process outlined in MIL-STD-882E. The appropriate level of rigor should be applied to the code that is generated by the KEEL Toolkit to minimize possibility of a mishap.

(THIS PAGE INTENTIONALLY LEFT BLANK)

**DISTRIBUTION**

	<u>Copies Paper/CD</u>
<b>DOD ACTIVITIES (CONUS)</b>	
DEFENSE TECHNICAL INFORMATION CENTER 8725 JOHN J KINGMAN ROAD FT BELVOIR VA 22060-6218	0/2
<b>NON-DOD ACTIVITIES (CONUS)</b>	
ATTN DOCUMENT CENTER CNA 3003 WASHINGTON BOULEVARD ARLINGTON VA 22201	1/1
ATTN THOMAS M. KEELEY COMPSIM LLC P O BOX 532 BROOKFIELD WI 53008	1/1
ATTN GOVERNMENT DOCUMENTS SECTION LIBRARY OF CONGRESS 101 INDEPENDENCE AVENUE SE WASHINGTON DC 20540-4172	4/4
<b>INTERNAL</b>	
1033 (TECHNICAL LIBRARY)	2/2
R44 (ABID MEHMOOD)	1/1
R44 (GUNENDRAN SIVAPRAGASAM)	4/4





